DRAM Caching for Remote Memory

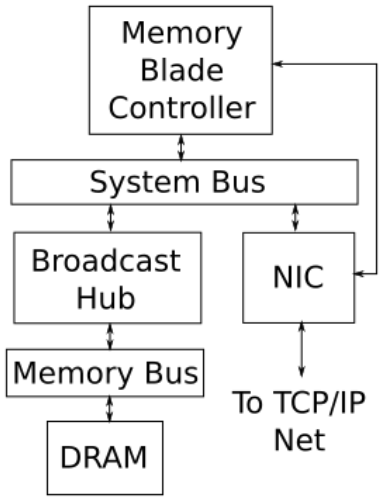
Howard Mao

Motivation

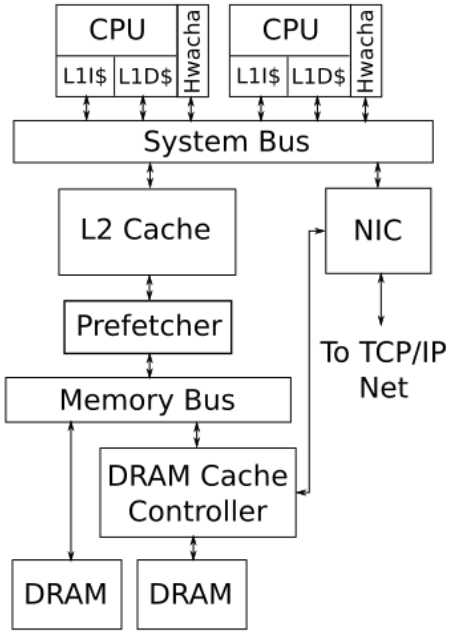
- Need to improve utilization of memory for data center applications
- Disaggregation is a known strategy for improving resource utilization
- Existing disaggregated memory systems
  - RDMA - complex API; low overhead
  - Page swapping - simple API; high overhead
- Local DRAM as hardware-managed LLC gets both
  - Transparent to software
  - Hardware-managed cache refill much lower overhead than page fault
  - Even lower overhead if we add hardware prefetching

System-Level Design

Memory Blade



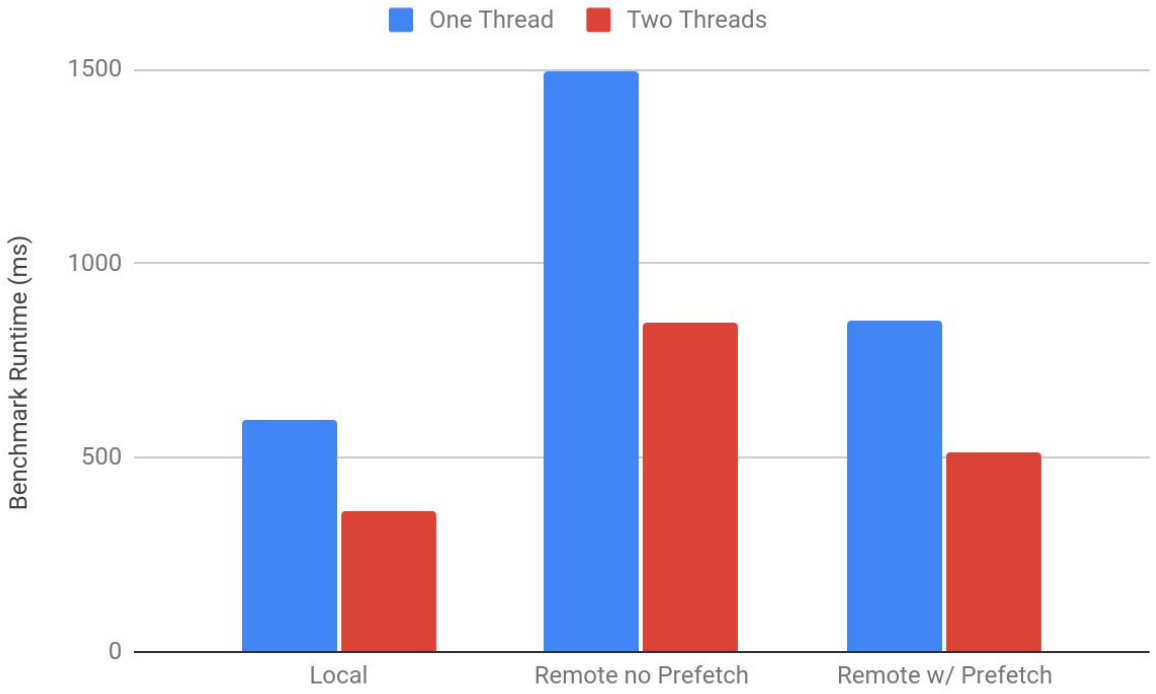
Compute Blade
w/ DRAM Cache



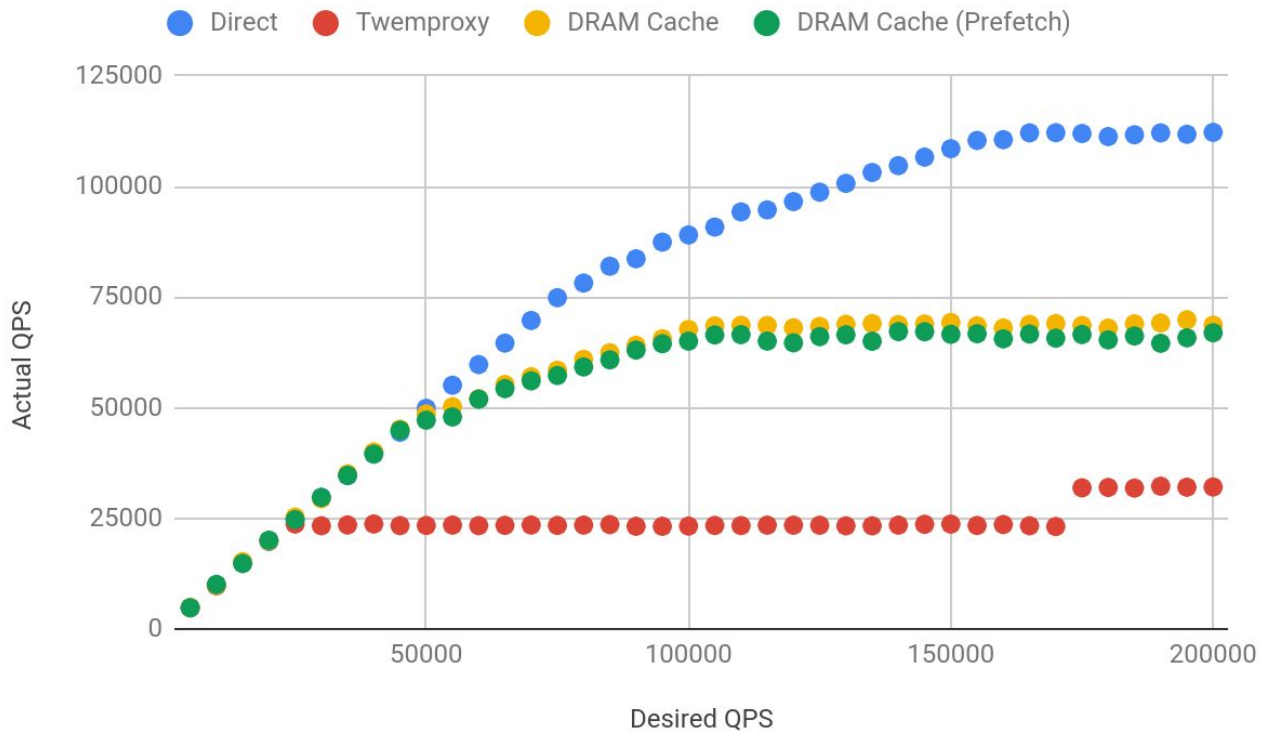
Evaluation Strategy

- Build RTL models of memory blade and compute blade using Chipyard
- Use FireSim to simulate multi-node systems
- Test both batch and interactive workloads
  - Batch: graph algorithm (friend of friends)
  - Interactive: memcached

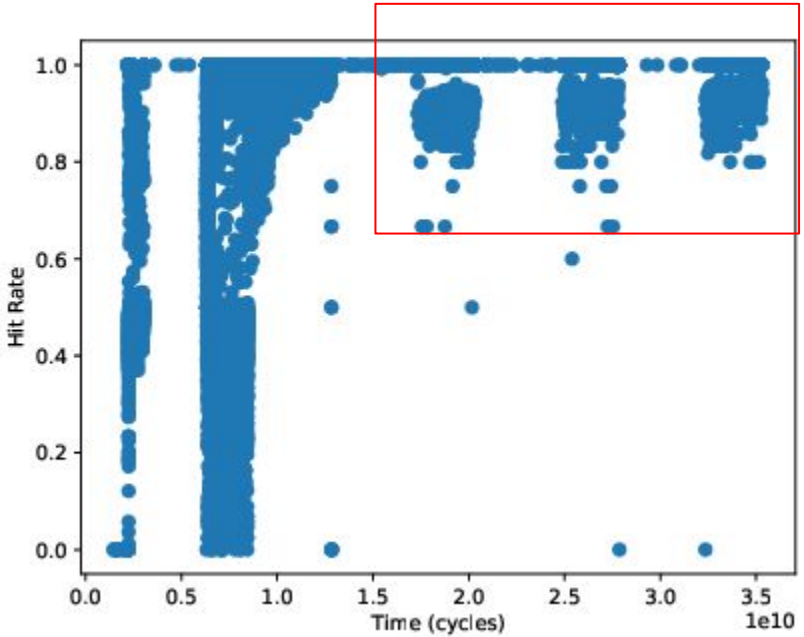
Friends of Friends Results



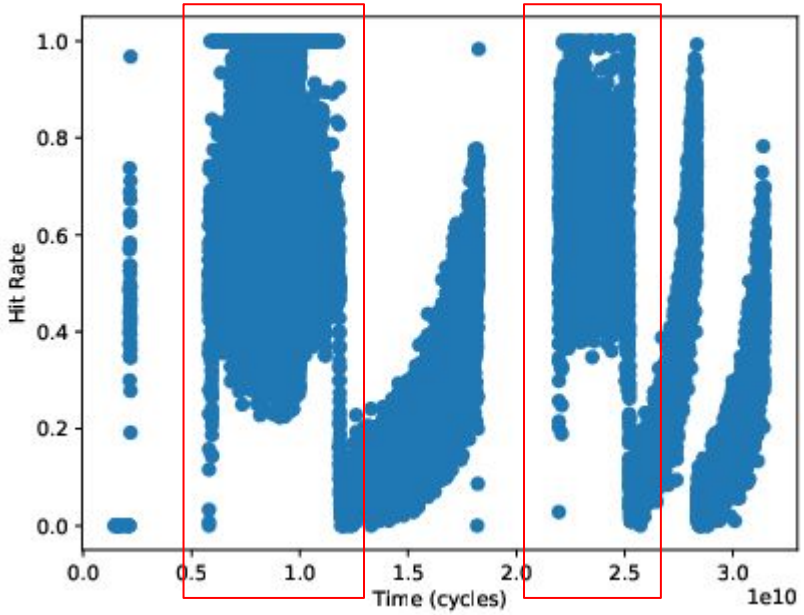
Memcached QPS



Memcached Hit Rate



Friends of Friends Hit Rate



Conclusion

- DRAM Cache well-suited to batch workloads
  - Automatic prefetching works quite well given sufficient spatial locality
  - Can easily expand backing memory by mapping more memory blades
- Less well-suited to interactive applications
  - Good temporal locality can make up for longer latencies up to a point
  - Decent substitute for software-based proxies / load balancers
- FireSim is a great platform for performance evaluation and debugging
  - Use real RTL, not software microarch model
  - Golden Gate transforms allow cycle-accurate timing
  - AWS and switch model allows simulation to scale out
  - Synthesized printf allows out-of-band logging and stats collection